NESTAR SYSTEMS, INCORPORATED
CLUSTER/ONE MODEL A
NETWORK SUPPORT PROGRAMS

NETWORK INTERFACE ROUTINES

Version 1.Ø, August, 198Ø

DESCRIPTION

There are now several Nestar provided pascal routines (units, actually) for sending and receiving messages over the Nestar Cluster/One Model A network.

The routines provide access to the network at various 'levels'; the 'high' level routines encapsulate most details of the interface while the 'low' level routines give more control over message handling.

The highest level routine is NFSCMD (formerly called SENDMSG), which takes a string containing a command for the Nestar File Server and returns a string containing the status line from NFS (typically "Ø,OK".) Any associated messages from NFS are displayed directly on the screen, not returned to the program. Thus, NFSCMD(Ø,'SHOW DATE',response) displays the date on the screen and returns "Ø,OK" in the string named 'response'.

The next level routine is NFSCMD1, which sends a command to the file server but retains control over all associated reply messages. NFSCMD1 is called repeatedly, each time returning the next line of text from NFS, until the final message from NFS (e.g. "Ø,OK" or whatever) is received.

The lowest level routines are BUSSEND and BUSRCV, which send and receive single messages to and from any network station (not restricted to the file server.)

OPERATING INSTRUCTIONS

A new library, PASCAL:NETWORK.LIBRARY, contains the routines. To use any of the routines your program should include:

```
  USES (*$U PASCAL:NETWORK.LIBRARY *) unitname;
```

where "unitname" in the USES instruction should be "NFSUNIT" for NFSMD, "NFSUNIT1" for NFSCMD1, and "BUSUNIT" for BUSSEND and BUSRCV. The object file must be linked using PASCAL:NETWORK.LIBRARY as one of the "lib file"'s.

Specific instructions for each of the routines, including listings of the unit interfaces and sample usage in programs, are given below.

Highest level interface - NFSCMD.

The unit NFSUNIT provides the function NFSCMD. The unit interface is as follows:

```
  UNIT NFSUNIT;

  INTERFACE

  TYPE NETMSG = STRING[255];

  FUNCTION NFSCMD (NETID: INTEGER; COMMAND: NETMSG;
                   VAR RETMSG: STRING):INTEGER;

  (* sends the given string (command) across the network    *)
  (* to NFS, and returns the final status message in the    *)
  (* string retmsg and the error code as the function value *)
  (* messages from NFS other than the status message are    *)
  (* displayed directly on the screen                       *)
  (* netid is Ø; reserved for future multiple networks      *)
```

A sample program which uses NFSCMD is :

```
  program minicmd;
  uses (*$U PASCAL:NETWORK.LIBRARY*) nfsunit;
  var cmd,reply : string;
      rc : integer;
  begin
    repeat
      write('ENTER CMD: ');
      readln(cmd);
      rc := nfscmd(Ø,cmd,reply);
      writeln(rc,',',reply);
    until cmd = 'QUIT';
  end. (*minicmd*)
```

<u>Intermediate level interface</u> - <u>NFSCMD1</u>.

The unit NFSUNIT provides the function NFSCMD. The unit interface is as follows:

```
UNIT NFSUNIT1;

INTERFACE

PROCEDURE NFSCMD1(NEWCMD:BOOLEAN; CMD:STRING;
                  VAR DONE: BOOLEAN; VAR RSP:STRING;
                  VAR ERROR:BOOLEAN; VAR ERRNUM:INTEGER);

(* use nfscmd1 only to intercept screen displayed    *)
(* responses.  use nfscmd for sending nfs commands   *)
(* unless it is necessary to intercept nfs responses *)
(* that are normally displayed directly              *)
(* on the local screen without program intervention. *)

(* called with newcmd = true for initially sending   *)
(* cmd to nfs.  then iterate calling nfstalk with    *)
(* newcmd=false until done = true or error=true.     *)
(* when error=true errnum gives network error number *)
(* detected.  when done = true rsp contains the last *)
(* text rec´d from network (eg Ø,OK).                *)
```

Code to use this unit looks like:

```
USES (*$U PASCAL:NETWORK.LIBRARY*) NFSUNIT1;
cmd := ´command´;
nfscmd1(true,cmd,done,res,error,errnum);
while not done and not error do
begin
  process RSP;
  nfscmd1(false,cmd,done,rsp,error,errnum);
end;
process final response from nfs if desired;
```

A sample program which uses NFSCMD1 is:

```
PROGRAM DIRLIST;

USES (*$U PASCAL:NETWORK.LIBRARY*) NFSUNIT1;
```

```
    VAR CMD, RSP : STRING; (*command to send to nfs*)
        DONE, ERROR : BOOLEAN;
        FILENAME, YORN : STRING;
        OUTFILE: FILE OF CHAR;
        I, INDX, ERRNUM : INTEGER;
        BUFFER : ARRAY[Ø..4ØØ] OF STRING[4Ø];


     BEGIN
       INDX := Ø;

       WRITE ('ENTER DIRECTORY PATHNAME TO BE LISTED:   ');
       READLN (CMD);
       INSERT('LIST ',CMD,1);

       WRITE ('NESTED LISTING? (Y/N) ');READLN(YORN);
       IF (YORN ='Y') OR (YORN='y')
       THEN INSERT(',NESTED',CMD,LENGTH(CMD)+1);

       WRITE ('VERBOSE LISTING? (Y/N) ');READLN(YORN);
       IF (YORN ='Y') OR (YORN='y')
       THEN INSERT(',VERBOSE',CMD,LENGTH(CMD)+1);

       WRITE ('FILENAME TO WRITE DIRECTORY LISTING TO?   ');
       READLN(FILENAME);
       IF      (POS('.TEXT',FILENAME)=Ø)
          AND  (POS('.text',FILENAME)=Ø)
          AND  (FILENAME[LENGTH(FILENAME)] <> '.')
         THEN  INSERT('.TEXT',FILENAME,LENGTH(FILENAME)+1);
       REWRITE(OUTFILE,FILENAME);

       NFSCMD1(TRUE,CMD,DONE,BUFFER[INDX],ERROR,ERRNUM);
       WHILE NOT DONE AND NOT ERROR DO
       BEGIN
         WRITE(BUFFER[INDX]);
         INDX := INDX+1;
         NFSCMD1(FALSE,CMD,DONE,BUFFER[INDX],ERROR,ERRNUM);
       END;
       WRITELN(BUFFER[INDX]);
       FOR I:=Ø TO INDX-1 DO WRITE(OUTFILE,BUFFER[I]);
       IF NOT ERROR THEN CLOSE(OUTFILE,LOCK);
     END. (*dirlist*)
```

Low level interface - BUSSEND and BUSRCV.

The unit BUSUNIT provides the functions BUSSEND and BUSRCV.

The unit interface is as follows:

```
UNIT BUSUNIT;

INTERFACE

VAR ME : INTEGER; (*my station number*)

FUNCTION BUSSEND(MSGTYPE,LEN,STN,MSGADDR:INTEGER):INTEGER;

(* Sends message at address MSGADDR of length LEN bytes     *)
(* of type MSGTYPE (message type is interpreted by the      *)
(* sender and receiver by mutual convention, however        *)
(* message types Ø to 127 are reserved by Nestar, their     *)
(* use may cause unpredicable results)                      *)
(* to station STN.  function value is network return code   *)
(* (bussend = Ø if message transmitted ok)                  *)

FUNCTION BUSRCV
  (LEN, MSGADDR:INTEGER;
  VAR MSGTYPE,TLEN,STN:INTEGER):INTEGER;

(* Receives message into address MSGADDR      *)
(* of maximum length LEN bytes                *)
(* (TLEN = actual length rec´d)               *)
(* of type MSGTYPE from station STN.          *)
(* Function value is network return code      *)
(* (busrcv = Ø if message transmitted ok)     *)
```

The following is a sample program for exercising the routines:

```
program bustest;

uses (*$U pascal:network.library*) busunit;
const maxtries = 5;
type longstring = string[255];
var str:↑longstring;
    mtyp, stn, tlen, result:integer;
    i, tries : integer;


begin
  new(str);
  repeat
```

```
    write('send, recv, or quit?  (r/s/q) ');readln(str↑);
    if (str↑ = 's') or (str↑ = 'S') then
    begin
      write('msg?  ');readln(str↑);
      write('to?   ');readln(stn);
      tries:=Ø; result := -1;
      while (result <> Ø) and (tries < maxtries) do
      begin

        (*  FUNCTION BUSSEND                          *)
        (* (MSGTYPE,LEN,STN,MSGADDR:INTEGER):INTEGER; *)

        result := bussend(6,length(str↑)+1,stn,ord(str));
        if result <> Ø then
        begin
          tries := tries + 1;
          if tries = 1 then write('retrying')
          else write('.');
          for I:=1 to 1ØØØ (*built in delay*) do;
        end
        else if tries > Ø then writeln;
      end; (*while*)
      if result <> Ø then begin
        writeln('.station ',stn,' not listening');
      end;
    end
    else if (str↑ = 'r') or (str↑ = 'R') then
    begin
      str↑ := '';
      (*  FUNCTION BUSRCV(LEN, MSGADDR:INTEGER;
             VAR MSGTYPE,TLEN,STN:INTEGER):INTEGER; *)
      result := busrcv(255,ord(str),mtyp,tlen,stn);
      if result = Ø
        then writeln('msg is ',str↑,' from ',stn)
        else writeln('result is ',result);
    end
    else if (str↑ = 'q') or (str↑ = 'Q')
      then exit(program);
  until false
end. (*bustest*)
```